



'(11) Publication number: **07175650 A**

Generated Document.

(21) Application number: 05317876

(51) Intl. Cl.: G06F 9/32 G06F 9/32 G06F 9/38

(22) Application date: 17.12.93

(30) Priority:

(43) Date of application publication: 14.07.95

(84) Designated contracting states:

(71)
Applicant: TOSHIBA CORP

(72) Inventor: MAEDA RIKA
TAKEUCHI YOICHIRO
SUZUKI SHINICHIRO
MORI YOSHIYA
TAKEUCHI KAZUAKI

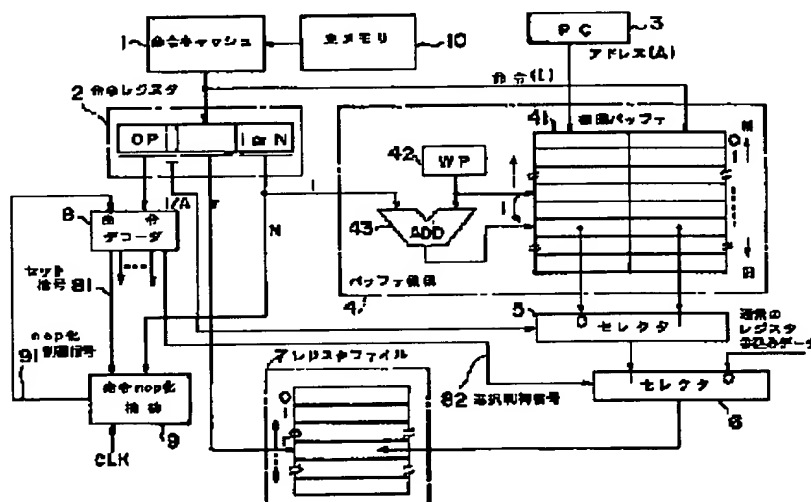
(74)
Representative:

(54) ARITHMETIC PROCESSOR

(57) Abstract:

PURPOSE: To enable conditional processing with no branching, to easily obtain an instruction or an address before a designated cycle and to easily perform immediate data extraction or argument exchange.

CONSTITUTION: Each time an instruction is fetched into an instruction register 2, the instruction concerned and the address are successively stored in a circulation buffer 41. This fetched instruction is decoded by an instruction decoder 8. When this instruction is a SKIP(N) instruction, an instruction nop processing control signal 91 is outputted from an instruction nop processing mechanism 9 to the instruction decoder 8 just for the period of a cycle shown by a value N of the N field and during that period, the decode object instruction is processed as an nop instruction. When the instruction is a GET(i, r) or GETA(i, r) instruction, among the instructions and addresses stored in the circulation buffer 41 before the cycle shown by a value (i) of the (i) field, the instruction or the address designated by the I/A bits is extracted to a register inside a register file 7 designated by the (r) field.



【特許請求の範囲】

【請求項 1】 命令群が記憶された記憶手段と、
前記記憶手段から取出された命令をデコードし、同命令
を実行するのに必要な各種デコード信号を出力する命令
デコード手段であって、当該命令が後続の命令を指定命
令数だけノー・オペレーション化する指定命令数ノー・
オペレーション化命令の場合に、対応する特定デコード
信号を出力する命令デコード手段と、
前記命令デコード手段から前記特定デコード信号が出力
された場合、当該命令デコード手段によってデコードさ
れている前記指定命令数ノー・オペレーション化命令の
指定する命令数に相当するサイクルの期間中、その間に
デコードされる後続の命令をノー・オペレーション命令
として扱うためのノー・オペレーション化制御信号を前
記命令デコード手段に出力する命令ノー・オペレーショ
ン化手段とを具備し、前記命令デコード手段は、前記命
令ノー・オペレーション化手段から前記ノー・オペレー
ション化制御信号が出力されている期間、デコード対象
となる命令を、その命令種別に無関係にノー・オペレー
ション命令として扱うことを特徴とする演算処理装置。

【請求項 2】 前期記憶手段に、条件が成立したときのみ
実行する命令群をノー・オペレーション化するための前
記指定命令数ノー・オペレーション化命令を、この命令
群の命令の直前に挿入して記憶しておき、当該ノー・オ
ペレーション化命令が実行された場合には、当該ノー・オ
ペレーション化命令で指定される数の命令がノーオペ
レーション化されるようにしたことを特徴とする請求項
1 記載の演算処理装置。

【請求項 3】 命令群が記憶された記憶手段と、
前記記憶手段から取出された命令をデコードし、同命令
を実行するのに必要な各種デコード信号を出力する命令
デコード手段であって、当該命令が後続の命令を指定命
令数だけノー・オペレーション化する指定命令数ノー・
オペレーション化命令の場合には、対応する第 1 のデコ
ード信号を出力し、当該命令が指定サイクル前の命令を
取出す命令履歴取出し命令の場合には、対応する第 2 の
デコード信号を出力する命令デコード手段と、
前記デコード手段によるデコードの対象となる命令を順
次格納するためのバッファ手段と、
このバッファ手段における前記命令の格納先を順次切
換え指定する書込みポインタ手段と、
前記命令デコード手段から前記第 1 のデコード信号が出
力された場合、当該命令デコード手段によってデコード
されている前記指定命令数ノー・オペレーション化命令
の指定する命令数に相当するサイクルの期間中、その間
にデコードされる後続の命令をノー・オペレーション命
令として扱うためのノー・オペレーション化制御信号を
前記命令デコード手段に出力する命令ノー・オペレーシ
ョン化手段と、
前記命令デコード手段から前記第 2 のデコード信号が出

力されている場合、当該命令デコード手段によってデコ
ードされている前記命令履歴取出し命令の指定するサイ
クル数と前記書込みポインタ手段の指す前記バッファ手
段内位置をもとに、このバッファ手段から指定サイク
ル前の命令を取得する手段とを具備し、前記命令デコー
ド手段は、前記命令ノー・オペレーション化手段から前記
ノー・オペレーション化制御信号が出力されている期
間、デコード対象となる命令を、その命令種別に無関係
にノー・オペレーション命令として扱うことを特徴とす
る演算処理装置。

【請求項 4】 命令群が記憶された記憶手段と、
前記記憶手段から取出された命令をデコードし、同命令
を実行するのに必要な各種デコード信号を出力する命令
デコード手段であって、当該命令が後続の命令を指定命
令数だけノー・オペレーション化する指定命令数ノー・
オペレーション化命令の場合には、対応する第 1 のデコ
ード信号を出力し、当該命令が指定サイクル前の命令の
アドレスを取出すアドレス履歴取出し命令の場合には、
対応する第 2 のデコード信号を出力する命令デコード手
段と、

前記デコード手段によるデコードの対象となる命令のア
ドレスを順次格納するためのバッファ手段と、
このバッファ手段における前記アドレスの格納先を順次
切換え指定する書込みポインタ手段と、
前記命令デコード手段から前記第 1 のデコード信号が出
力された場合、当該命令デコード手段によってデコード
されている前記指定命令数ノー・オペレーション化命令
の指定する命令数に相当するサイクルの期間中、その間
にデコードされる後続の命令をノー・オペレーション命
令として扱うためのノー・オペレーション化制御信号を
前記命令デコード手段に出力する命令ノー・オペレーシ
ョン化手段と、
前記命令デコード手段から前記第 2 のデコード信号が出
力されている場合、当該命令デコード手段によってデコ
ードされている前記アドレス履歴取出し命令の指定する
サイクル数と前記書込みポインタ手段の指す前記バッ
ファ手段内位置をもとに、このバッファ手段から指定サイ
クル前の命令のアドレスを取得する手段とを具備し、前
記命令デコード手段は、前記命令ノー・オペレーション
化手段から前記ノー・オペレーション化制御信号が出力
されている期間、デコード対象となる命令を、その命令
種別に無関係にノー・オペレーション命令として扱うこ
とを特徴とする演算処理装置。

【請求項 5】 命令群が記憶された記憶手段と、
前記記憶手段から取出された命令をデコードし、同命令
を実行するのに必要な各種デコード信号を出力する命令
デコード手段であって、当該命令が後続の命令を指定命
令数だけノー・オペレーション化する指定命令数ノー・
オペレーション化命令の場合には、対応する第 1 のデコ
ード信号を出力し、当該命令が指定サイクル前の命令を

取出す命令履歴取出し命令または指定サイクル前の命令のアドレスを取出すアドレス履歴取出し命令の場合には、対応する第 2 のデコード信号を出力する命令デコード手段と、

前記デコード手段によるデコードの対象となる命令およびそのアドレスを順次格納するためのバッファ手段と、このバッファ手段における前記命令およびアドレスの格納先を順次切換え指定する書込みポインタ手段と、前記命令デコード手段から前記第 1 のデコード信号が出力された場合、当該命令デコード手段によってデコードされている前記指定命令数ノー・オペレーション化命令の指定する命令数に相当するサイクルの期間中、その間にデコードされる後続の命令をノー・オペレーション命令として扱うためのノー・オペレーション化制御信号を前記命令デコード手段に出力する命令ノー・オペレーション化手段と、

前記命令デコード手段から前記第 2 のデコード信号が出力されている場合、当該命令デコード手段によってデコードされている前記命令履歴取出し命令またはアドレス履歴取出し命令の指定するサイクル数と前記書込みポインタ手段の指す前記バッファ手段内位置をもとに、このバッファ手段から指定サイクル前の命令またはアドレスを取得する手段とを具備し、前記命令デコード手段は、前記命令ノー・オペレーション化手段から前記ノー・オペレーション化制御信号が出力されている期間、デコード対象となる命令を、その命令種別に無関係にノー・オペレーション命令として扱うことを特徴とする演算処理装置。

【請求項 6】前期記憶手段に、前記指定命令数ノー・オペレーション命令、当該命令によりノー・オペレーション化される命令部分に設定される即値データ、および取得したい即値データを命令と見なしてサイクル数を指定することで前記バッファ手段から取出すための前記命令履歴取出し命令を記憶しておき、当該命令履歴取出し命令が実行された場合に、当該命令履歴取出し命令の指定するサイクル前の前記即値データを前記バッファ手段から取得するようにしたことを特徴とする請求項 3 または請求項 5 記載の演算処理装置。

【請求項 7】前期記憶手段に、前記指定命令数ノー・オペレーション命令、当該命令によりノー・オペレーション化される命令部分に設定される手続き呼出しの引数、および取得したい引数を命令と見なしてサイクル数を指定することで前記バッファ手段から取出すための前記命令履歴取出し命令を記憶しておき、当該取出し命令が実行された場合に、当該取出し命令の指定するサイクル前の前記引数を前記バッファ手段から取得するようにしたことを特徴とする請求項 3 または請求項 5 記載の演算処理装置。

【請求項 8】前期記憶手段内の、実行開始サイクルより N サイクル後にエラートラップの発生する可能性のある

命令が実際にエラートラップした場合の飛び先に、サイクル数 N+1 を指定する前記アドレス履歴取出し命令を記憶しておき、当該取出し命令が実行された場合に、当該取出し命令の指定するサイクル前の前記エラートラップした命令のアドレスを前記バッファ手段から取得するようにしたことを特徴とする請求項 4 または請求項 5 記載の演算処理装置。

【請求項 9】前記記憶手段内の、手続き呼出し命令の飛び先に、サイクル数 1 を指定する前記アドレス履歴取出し命令を記憶しておき、当該取出し命令が実行された場合に、当該取出し命令の指定するサイクル前の前記手続き命令のアドレスを前記バッファ手段から取得し、そのアドレスの次のアドレスを戻りアドレスとして取得することを特徴とする請求項 4 または請求項 5 記載の演算処理装置。

【発明の詳細な説明】

【0001】

【産業上の利用分野】この発明は、後続の命令を指定命令数だけ nop (ノー・オペレーション) 化する専用命令を含む命令列を実行する演算処理装置に関する。

【0002】

【従来の技術】一般に、演算処理装置で条件処理（条件分岐処理）を実行可能とするためには、分岐先の命令にラベルを付けておくのが一般的である。このような条件処理のための命令列について、図 8 を参照して説明する。

【0003】まず、図 8 (a) は、n の値により異なる命令処理を実行するためのソースコードの一例を示すもので、n=0 のときは命令 # 1 を、n=1 のときは命令 # 2 を、そして n=2 のときは命令 # 3 をそれぞれ実行して、例えば共通の命令処理に進む条件処理を指定している。ここでの条件処理は、例えばエラー要因 n (0~2) によって、その要因 n (0~2) 別にトラップを発生させる命令 # 1~# 3 に分岐する場合であるものとする。

【0004】さて、図 8 (a) に示すようなソースコードの指定する条件処理を、演算処理装置で実行可能とするためには、同ソースコードをコンパイルしてオブジェクトコード（ロードモジュール）を作成する必要がある。従来、条件処理は、条件分岐命令と分岐先の命令との組合せにより実現されることから、図 8 (a) に示すソースコードは、図 8 (b) に示すようにコンパイルされるのが一般的である。図中 SUB (n-0), SUB (n-1), SUB (n-2) は、それぞれ (n-0), (n-1), (n-2) の減算を指定する減算命令 (SUB 命令)、JNZ (L1), JNZ (L2), JNZ (L3) は、先行する命令 (SUB 命令) の演算結果がゼロでない場合に、ラベル L1, L2, L3 が付された命令 (分岐先命令) にジャンプすることを示す条件分岐命令、JP (L100) は、L100 の命令に無

条件でジャンプするジャンプ命令である。

【0005】このように、従来の演算処理装置で条件処理を実現するには、分岐先の命令にラベルを付けておかなければならず、レジスタを用いた分岐先アドレスの計算等が必要となる他、条件（上記の例ではnの値）が複数の場合には、別々に条件判定を行うことから、命令数が増加するといった問題があった。

【0006】また、エラー処理の自動化を行うためのエラー情報解析処理などにおいては、指定サイクル前の実行命令を知りたい場合が発生する。ところが従来の演算処理装置では、途中で条件処理が存在すると目的の命令を見つけることは不可能であり、人がロードモジュールをたどって見つけなければならなかった。

【0007】また従来は、即値データ（即値オペランドデータ）や引数は、レジスタからのデータロードにより得る必要があった。更に、戻りアドレスをリンクレジスタに設定しておく必要があった。

【0008】

【発明が解決しようとする課題】上記したように従来の演算処理装置では、条件処理を実現するのに、分岐先の命令にラベルを付けておかなければならず、レジスタを用いた分岐先アドレスの計算等が必要となる他、命令数が増加するといった問題があった。

【0009】また、指定サイクル前の実行命令を見つけるには、人がロードモジュールをたどらなければならないという問題もあった。また、レジスタからのデータロードによるメモリのオーバーヘッドやレジスタを使用することによるレジスタ増に伴うコストの増加を招くという問題もあった。

【0010】この発明は上記事情を考慮してなされたものでその目的は、分岐なしの条件処理が実現でき、もってレジスタを用いた分岐アドレスの計算等を不要とする他、命令数を削減できる演算処理装置を提供することにある。

【0011】この発明の他の目的は、指定サイクル前の命令またはアドレスの少なくとも一方が簡単に取得できる演算処理装置を提供することにある。この発明の更に他の目的は、即値データの取出しや引数の受け渡しがレジスタからのロードを必要とせずに簡単に行える演算処理装置を提供することにある。この発明の更に他の目的は、手続き呼出し時の戻りアドレスが、リンクレジスタへの設定操作を必要とせずに取得できる演算処理装置を提供することにある。

【0012】

【課題を解決するための手段および作用】この発明は、命令デコード手段によりデコードされている命令が後続の命令を指定命令数だけノー・オペレーション（nop）化する指定命令数ノー・オペレーション化命令（SKIP(N)命令）の場合に、当該命令デコード手段から特定デコード信号（第1のデコード信号）が出力され

る構成とすると共に、当該命令デコード手段から特定デコード信号（第1のデコード信号）が出力された場合、当該命令デコード手段によってデコードされているSKIP(N)命令の指定する命令数Nに相当するNサイクルの期間中、その間にデコードされる後続の命令をnop命令として扱うためのnop化制御信号を命令デコード手段に出力する命令nop化手段（命令ノー・オペレーション化手段）を設けたことを特徴とするものである。

【0013】このような構成においては、命令デコード手段および命令nop化手段の組合せ動作により、SKIP(N)命令で指定された飛ばしたい個数だけ後続の命令がnop化されるため、分岐なしで条件処理を実現することが可能となる。

【0014】また、この発明は、上記命令デコード手段によりデコードされる命令およびそのアドレスを順次格納するためのバッファ手段と、このバッファ手段における上記命令およびアドレスの格納先を順次切換え指定する書込みポインタ手段とを更に設ける他、上記命令デコード手段によりデコードされている命令が、指定サイクル前の命令を取出す命令履歴取出し命令（GETI(i, r)命令）または指定サイクル前の命令のアドレスを取出すアドレス履歴取出し命令（GETA(i, r)命令）の場合には、当該命令デコード手段から第2のデコード信号が出力される構成とし、この第2のデコード信号が出力されている場合、命令デコード手段によってデコードされているGETI(i, r)命令またはGETA(i, r)命令の指定するサイクル数iと書込みポインタ手段の指すバッファ手段内位置をもとに、このバッファ手段からiサイクル前の命令またはアドレスを取得するようにしたことを特徴とする。

【0015】このような構成においては、命令デコード手段によりデコードされる命令およびそのアドレスがバッファ手段に順次格納される。そして、命令デコード手段によりGETI(i, r)またはGETA(i, r)命令がデコードされた場合には、バッファ手段に格納されている情報のうち、GETI(i, r)またはGETA(i, r)命令中のiフィールドの指定するサイクル(iサイクル)前の命令またはそのアドレスが、例えばGETI(i, r)またはGETA(i, r)命令中のrフィールドの指定するレジスタファイル内レジスタに取出される。

【0016】したがって、履歴を取得したい命令よりiサイクル後に実行される位置にGETI(i, r)命令を用意しておくことで、当該命令GETI(i, r)命令が実行された場合には、当該GETI(i, r)命令中のiフィールドの値で指定されたiサイクル前の命令を命令履歴として、例えば当該GETI(i, r)命令中のrフィールドの指定するレジスタファイル内レジスタに取出すことが可能となる。

【0017】同様に、SKIP(N)命令によりnop化される命令部分に、即値データ(即値オペランドデータ)または手続き呼出しの固定引数を設定しておき、更にその後GETI(i, r)命令を用意して、これらを実行させることにより、バッファ手段に格納されている情報のうち、当該GETI(i, r)命令中のiフィールドの値でiサイクル前の命令として指定された即値データまたは固定引数を、例えば当該GETI(i, r)命令中のrフィールドの指定するレジスタファイル内レジスタに取出すことが可能となる。

【0018】同様に、実行開始サイクルよりNサイクル後にエラートラップの発生する可能性のある命令(対象命令)が実際にエラートラップした場合の飛び先に、サイクル数N+1、即ちi=N+1を指定するGETA(i, r)命令を用意しておくことで、上記対象命令がエラートラップして当該GETA(i, r)命令に実行が移った場合に、バッファ手段に格納されている情報のうち、当該GETA(i, r)命令中のiフィールドの値で指定されたN+1サイクル前の命令のアドレス、即ち上記対象命令のアドレスをエラー発生命令のアドレスとして、例えば当該GETA(i, r)命令中のrフィールドの指定するレジスタファイル内レジスタに取出すことが可能となる。

【0019】同様に、手続き呼出し命令の飛び先に、サイクル数1を指定するGETA(i, r)命令を用意しておくことで、当該GETA(i, r)が呼出された場合に、バッファ手段に格納されている情報のうち、当該GETA(i, r)命令中のiフィールドの値で指定された1サイクル前の命令のアドレス、即ち上記手続き呼出し命令のアドレスを、戻りアドレスの1つ前のアドレスとして、例えば当該GETA(i, r)命令中のrフィールドの指定するレジスタファイル内レジスタに取出すことが可能となる。

【0020】

【実施例】以下、この発明をパイプライン制御方式の演算処理装置に適用した一実施例につき、図面を参照して説明する。図1は同実施例に係る演算処理装置の主要部の構成を示すブロック図である。

【0021】同図において、1は後述する主メモリ10に格納されている命令群の一部の写しが置かれる命令キャッシュである。この命令キャッシュ1に実行すべき主メモリ10上の命令が存在しないミスヒット時には、当該命令が主メモリ10から命令キャッシュ1に読込まれて使用される。なお、命令キャッシュ1は、命令取出しの高速化のために設けられるもので、必ずしも必要ではない。

【0022】2は命令キャッシュ1から取出された(フェッチされた)命令が保持される命令レジスタである。ここで、本実施例で適用される新規な命令について図2を参照して説明する。

【0023】この新規な命令は、指定命令数nop化命令、命令履歴取出し命令およびアドレス履歴取出し命令の3種である。まず、指定命令数nop化命令は、後続の命令を指定命令数だけnop化することを指定するもので、図2(a)に示すように指定命令数nop化命令であることを示すOPコード・フィールド(オペレーション・コード・フィールド)、およびnop化する命令数Nを指定するフィールド(Nフィールド)とを有する。この指定命令数nop化命令をSKIPまたはSKIP(N)のように表現する。

【0024】次に命令履歴取出し命令は、指定サイクル前の命令を後述する循環バッファ41から取出すことを指定するもので、図2(b)に示すように、命令履歴取出し命令であることを示すOPコード・フィールド、取出した命令の格納先レジスタを指定するレジスタ(デスティネーション・レジスタ)指定フィールド(rフィールド)、および指定サイクル数iが設定される指定サイクル数フィールド(iフィールド)を有する。この命令履歴取出し命令をGETIまたはGETI(i, r)のように表現する。なお、上記iフィールドの位置およびサイズは、指定命令数nop化命令(SKIP(N)命令)のNフィールドのそれに一致する。

【0025】次にアドレス履歴取出し命令は、指定サイクル前のアドレスを循環バッファ41から取出すことを指定するもので、命令履歴取出し命令と同様の形式であり、図2(c)に示すように、アドレス履歴取出し命令であることを示すOPコード・フィールド、取出したアドレスの格納先レジスタを指定するレジスタ(デスティネーション・レジスタ)指定フィールド(rフィールド)、および指定サイクル数iが設定される指定サイクル数フィールド(iフィールド)を有する。このアドレス履歴取出し命令をGETAまたはGETA(i, r)のように表現する。

【0026】GETI(i, r)命令およびGETA(i, r)命令のOPコード・フィールドの例えば最下位ビットはI/Aビットとして位置付けられている。GETI(i, r)とGETA(i, r)の相違は、このI/Aビットの値が異なる点であり、GETI(i, r)のI/Aビットは“1”、GETA(i, r)のI/Aビットは“0”である。この逆としても構わないことは勿論である。

【0027】再び図1を参照すると、3は(命令キャッシュ1から)命令レジスタ2にフェッチされる命令の(主メモリ10上の)アドレスを指定するプログラムカウンタ(PC)である。プログラムカウンタ3は、各命令フェッチサイクル毎に、インクリメントされる。但し、ジャンプ等が発生した場合には、プログラムカウンタ3の内容は、ジャンプ先のアドレスに更新される。

【0028】4は命令キャッシュ1から命令レジスタ2に取出された命令と、その際のプログラムカウンタ3の

10

20

30

40

50

値（アドレス）の対を、各命令フェッチサイクル毎に順次保持するためのバッファ機構である。

【0029】バッファ機構4は、各エントリがサイクリックに使用されるバッファ（循環バッファ）41、この循環バッファ41内の命令・アドレスの書き込み先エントリを指す書き込み先ポインタ（WP）42、および加算器（ADD）43を有している。加算器43は、後述する命令デコーダ8によりデコードされている命令がGETI（i, r）命令またはGETA（i, r）命令の場合に、その命令中のiフィールドの示すサイクル数と書き込み先ポインタ42の示す値との加算を行い、循環バッファ41内の読出し対象エントリの情報（読出しポインタ情報）を生成するのに用いられる。

【0030】5はバッファ機構4内の循環バッファ41から読出される命令およびアドレスのうちのいずれか一方を命令デコーダ8によりデコードされている命令のOPコード・フィールド中の最下位ビット（同命令がGETI（i, r）命令またはGETA（i, r）命令の場合にはI/Aビット）に応じて選択するセクタ、6はセクタ5の出力情報および通常のレジスタ書き込みデータのうちの一方を後述する選択制御信号82に応じて選択するセクタ、7はレジスタ群からなるレジスタファイルである。レジスタファイル7は、セクタ6の出力情報を保持するのに用いられる。

【0031】8は命令レジスタ2に保持された命令のOPコード・フィールドをデコードして、各部を制御する各種デコード信号（制御信号）を出力する命令デコーダである。この命令デコーダ8は、OPコード・フィールドの内容がSKIP（N）命令を示す場合には、同命令中のNフィールドの値（指定命令数）を後述する命令nop化機構9内に設定するためのセット信号81を出力し、GETI（i, r）命令またはGETA（i, r）命令の場合には、セクタ6に対する選択制御信号82を出力するように構成されている。また命令デコーダ8は、命令nop化機構9から命令nop化制御信号91が出力されている期間中は、デコード対象となる命令を、その命令種別に無関係にnop命令として扱う。

【0032】9は命令nop化機構である。この命令nop化機構9は、命令デコーダ8からセット信号81が出力された場合に、同デコーダ8によりデコードされているSKIP（N）命令中のNフィールドの値Nを内部に設定し、その値Nの示すサイクルの期間（Nサイクルの期間）だけ、命令nop化制御信号91を出力する。命令nop化機構9は、例えば、命令デコーダ8のデコード対象となっているSKIP（N）命令中のNフィールドの値Nをセット信号81に応じてプリセットし、その値が「0」になるまでパイプライン・サイクルのクロックCLKに応じてダウンカウントするカウンタと、このカウンタの値が「0」でない期間、アクティブな命令nop化制御信号91を出力するゲート回路（いずれも

図示せず）により構成される。

【0033】10は命令列からなる各種プログラム、データ等が格納される主メモリである。次に、図1の構成の基本動作について、図3のフローチャート、および図4乃至図7の基本動作説明図を参照して説明する。

【0034】まず、プログラムカウンタ3の指定するアドレスの命令が命令キャッシュ1から取出され、命令レジスタ2に読込まれたものとする。同時に、バッファ機構4内の書き込み先ポインタ42が次の循環バッファ41内エントリを指すように更新され、命令キャッシュ1から取出された命令とプログラムカウンタ3の指定するアドレスの対が、当該更新後の書き込み先ポインタ42の指定する循環バッファ41内エントリに書込まれる（記録される）（ステップS1）。これにより、書き込み先ポインタ42は、命令レジスタ2にフェッチされた命令とそのアドレスを示すことになる。

【0035】この様子を、命令キャッシュ1から命令レジスタ2にフェッチされた命令が、主メモリ10上のアドレス“1000”のADD（加算）命令（ここではA+Bを指定するADD命令）である場合を例に、図4に示す。

【0036】なお、書き込み先ポインタ42は、循環バッファ41内エントリのアドレスが小さくなる方向に更新（デクリメント）されるものとする。また、書き込み先ポインタ42の値が「0」の状態を更新された場合、同ポインタ42の更新後の値はエントリアドレスの最大値を指すものとする。

【0037】命令レジスタ2にフェッチされた命令は、命令デコーダ8に導かれる。命令デコーダ8は、この命令のOPコード・フィールドの内容をデコードし、対応するデコード信号を出力する。

【0038】ここで、命令デコーダ8によりデコードされている命令が指定命令数nop化命令、即ちSKIP（N）命令の場合について説明する。この場合、命令デコーダ8からセット信号81が出力される。このセット信号81は、命令デコーダ8によりデコードされているSKIP（N）中のNフィールドの値と共に命令nop化機構9に導かれる。

【0039】すると命令nop化機構9は、命令デコーダ8によりデコードされているSKIP（N）中のNフィールドの値「N」を、セット信号81に応じて内部設定し、その値の示すサイクルの期間（Nサイクルの期間）だけ、アクティブな命令nop化制御信号91を出力する。

【0040】命令nop化機構9からの命令nop化制御信号91は命令デコーダ8に導かれる。すると命令デコーダ8は、命令nop化制御信号91がアクティブな期間（Nサイクルの期間）、命令レジスタ2を介して導かれる後続のN命令を、その命令種別に無関係にnop命令として扱う。

【0041】このように、図1の構成の演算処理装置では、命令デコーダ8にてデコードされている命令がSKIP(N)命令である場合(ステップS2)、後続のN命令がnop化される(ステップS3)。

【0042】したがって、主メモリ10上の或るアドレスに例えばN=2のSKIP(N)命令、即ちSKIP(2)命令が存在する場合には、図5に示すように、SKIP(2)命令に続く2つの命令がnop化されることになる。

【0043】次に、命令デコーダ8によりデコードされている命令が命令履歴取出し命令、即ちGETI(i, r)命令の場合について説明する。この場合、命令デコーダ8によりデコードされているGETI(i, r)命令中のiフィールドの値「i」と、その時点の書き込み先ポインタ42の値とが、加算器43により加算される。そして、加算器43の加算結果(である読出しポインタ情報)の指定する循環バッファ41内エントリに記録されている命令およびアドレスの対、即ち現在書き込み先ポインタ42の指定しているエントリ中の命令に対してiサイクル前の命令およびそのアドレスの対が、同バッファ41から読出される。

【0044】循環バッファ41から読出された命令およびアドレスはセクタ5に導かれる。このセクタ5には、命令デコーダ8によりデコードされているGETI(i, r)命令のOPコード・フィールド中のI/Aビットも(選択制御信号として)導かれる。このI/Aビットの値は、GETI(i, r)命令では“1”である。

【0045】セクタ5は、この例のようにI/A=1の場合、循環バッファ41から読出された命令およびアドレスのうちの命令をセクタ6に選択出力する。一方、命令デコーダ8は、デコード対象となっている命令がこの例のようにGETI(i, r)命令の場合、論理“1”の選択制御信号82を出力する。この選択制御信号82はセクタ6に導かれる。

【0046】セクタ6は、この例のように選択制御信号82が“1”の場合、セクタ5の選択出力情報(ここでは命令)を選択する。このように、図1の構成の演算処理装置では、命令デコーダ8にてデコードされている命令がGETI(i, r)命令である場合(ステップS4)、iサイクル前の命令およびそのアドレスの対が循環バッファ41から取出され、そのうちの命令がセクタ5、6を介してレジスタファイル7のrレジスタに取出される(ステップS5)。

【0047】この様子を、命令デコーダ8にてデコードされているGETI(i, r)命令のアドレスが“1000”であり、iサイクル前の命令がSUB(減算)命令(ここではC-Dを指定するSUB命令)である場合を例に、図6に示す。

【0048】次に、命令デコーダ8によりデコードされ

ている命令がアドレス履歴取出し命令、即ちGETA(i, r)命令の場合について説明する。この場合、命令デコーダ8によりデコードされているGETA(i, r)命令中のiフィールドの値「i」と、その時点の書き込み先ポインタ42の値とが、加算器43により加算される。そして、加算器43の加算結果(である読出しポインタ情報)の指定する循環バッファ41内エントリに記録されている命令およびアドレスの対が、即ち現在書き込み先ポインタ42の指定しているエントリ中の命令に対してiサイクル前の命令およびそのアドレスの対が、同バッファ41から読出される。

【0049】循環バッファ41から読出された命令およびアドレスはセクタ5に導かれる。このセクタ5には、命令デコーダ8によりデコードされているGETA(i, r)命令のOPコード・フィールド中のI/Aビットも(選択制御信号として)導かれる。このI/Aビットの値は、GETA(i, r)命令では“0”である。

【0050】セクタ5は、この例のようにI/A=0の場合、循環バッファ41から読出された命令およびアドレスのうちのアドレスをセクタ6に選択出力する。一方、命令デコーダ8は、デコード対象となっている命令がこの例のようにGETA(i, r)命令の場合、上記したGETI(i, r)命令のときと同様に論理“1”の選択制御信号82を出力する。

【0051】セクタ6は、この例のように選択制御信号82が“1”の場合、セクタ5の選択出力情報(ここではアドレス)を選択する。セクタ6により選択された情報(アドレス)は、この例のように、命令デコーダ8によりGETA(i, r)命令がデコードされている場合には、同命令中のrフィールドの内容「r」の指定するレジスタファイル7内レジスタ(rレジスタ)に格納される。

【0052】このように、図1の構成の演算処理装置では、命令デコーダ8にてデコードされている命令がGETA(i, r)命令である場合(ステップS6)、iサイクル前の命令およびそのアドレスの対が循環バッファ41から取出され、そのうちのアドレスがセクタ5、6を介してレジスタファイル7のrレジスタに取出される(ステップS7)。

【0053】この様子を、命令デコーダ8にてデコードされているGETA(i, r)命令のアドレスが“1050”であり、iサイクル前の命令のアドレスが“1000”である場合を例に、図7に示す。

【0054】次に、図1の演算処理装置において、SKIP命令(指定命令数nop化命令)を利用して、図8(a)に示したようなソースコードの指定する条件処理を、図8(c)に示すように分岐なしで実現した例につき、説明する。なお、図8(a)に示すソースコードは、[従来の技術]の欄で説明したように、n=0のと

きは命令#1を、 $n=1$ のときは命令#2を、そして $n=2$ のときは命令#3をそれぞれ実行して、例えば共通の命令処理に進む条件分岐処理を指定するものである。

【0055】さて、図8(a)に示すようなソースコードの指定する条件処理を、図1の構成の演算処理装置で実行可能とするためには、同ソースコードをコンパイルしてオブジェクトコード(ロードモジュール)を作成する必要がある。この際、条件が成立したときに実行する命令の前にSKIP命令を挿入してコンパイルする。本実施例では、図8(a)に示すソースコードをコンパイルした場合、図8(c)に示すように、SKIP($n*2$)、命令#1、SKIP(3)、命令#2、SKIP(1)および命令#3の順で続く命令列が作成される。

【0056】前記したように、図1の構成の演算処理装置においては、SKIP(N)命令により、後続のN命令がnop化される(図3ステップS2、S3)。したがって、図8(c)に示した命令列の例では、 $n=0$ の場合には、SKIP($n*2$)、即ちSKIP(0)に続いて、命令#1、SKIP(3)が順次実行され、更に後続の3命令(命令#2、SKIP(1)、命令#3)がnop化された後、命令#3の次の命令が実行される。

【0057】同様に、 $n=1$ の場合には、SKIP($n*2$)、即ちSKIP(2)が実行されて、後続の2命令(命令#1、SKIP(3))がnop化された後、命令#2、SKIP(1)が順次実行され、更に後続の1命令(命令#3)がnop化された後、命令#3の次の命令が実行される。

【0058】同様に、 $n=2$ の場合には、SKIP($n*2$)、即ちSKIP(4)が実行されて、後続の4命令(命令#1、SKIP(3)、命令#2、SKIP(1))がnop化された後、SKIP(1)の次の命令#3が実行され、更に次の当該命令#3の次の命令が実行される。

【0059】このように、図1に示す構成の演算処理装置では、SKIP命令(指定命令数nop化命令)を利用することで、分岐を用いずに条件処理を実現することができる。したがって、図8(c)から明らかなように、条件処理のための命令列の構成命令数が、図8

(b)に示した従来の場合の命令列に比較して削減できる。また、命令数が削減できることから、処理の高速化が図れ、更に分岐発生に伴うパイプライン停止の問題がないため、一層の高速化が図れる。

【0060】次に、図1の演算処理装置において、GETI命令(命令履歴取出し命令)を利用して、指定サイクル前の命令を取得する場合(命令履歴取出し)につき、図9を参照して説明する。

【0061】まず、GETI命令を利用して、指定サイクル前の命令を取得できるようにするには、途中で分岐がないようにしておく必要がある。このため、途中で条

件処理を必要とする場合には、先に述べたように、条件処理についてSKIP命令を用いた形にコンパイルすることで、分岐なしの条件処理を実現できるようにしておく。

【0062】図9は、主メモリ10上に、SKIP(1)、何らかの1命令、命令#1、何らかの2命令、GETI(3, r)の命令列が格納されている場合に、当該命令列が1命令ずつ順次実行され、GETI(3, r)が実行された際の様子を説明するためのものである。

【0063】図9の例では、SKIP(1)、何らかの1命令、命令#1、何らかの2命令、GETI(3, r)がSKIP(1)から順に1命令ずつ命令レジスタ2にフェッチされる毎に、書込み先ポインタ42がデクリメントされて、そのデクリメント後の当該ポインタ42の指定する循環バッファ41内エンタリに、上記フェッチされた命令とそのアドレスが記録される(ステップS1)。また、SKIP(1)の次の1命令は、当該SKIP(1)の指定によりnop化される(ステップS2, S3)。

【0064】さて、GETI(3, r)が命令レジスタ2にフェッチされて、当該GETI(3, r)とそのアドレスが、図9に示すように、書込み先ポインタ42の指す循環バッファ41内エンタリに記録され(ステップS1)、更に当該GETI(3, r)が命令デコーダ8でデコードされたものとする(ステップS4)。

【0065】この場合、図6を参照して説明したことからも明らかなように、GETI(3, r)のiフィールドの値「3」($i=3$)と書込み先ポインタ42の値との(加算器43による)加算値で指定される循環バッファ41内エンタリから読出される命令およびアドレスの対、即ち現在書込み先ポインタ42の指しているエンタリ中の命令GETI(3, r)に対して3サイクル前の命令およびアドレスの対のうちの命令(ここでは命令#1)が、図9に示すように、当該GETI(3, r)のrフィールドの指定する、レジスタファイル7内のrレジスタに格納される(ステップS5)。

【0066】ここで、SKIP(1)を用いることで条件処理を分岐なしで実現していることから、途中で条件処理があっても、GETI(3, r)の実行により、指定サイクル前(ここでは3サイクル前)の命令(命令#1)を、正しくレジスタファイル7内の指定レジスタ(rレジスタ)に取出すことができる。

【0067】次に、図1の演算処理装置において、SKIP命令(指定命令数nop化命令)でnop化された命令部分(位置)に即値データを格納しておき、当該即値データをGETI命令(命令履歴取出し命令)を利用してレジスタファイル7内に取出す場合(即値オペランド指定)につき、図10を参照して説明する。

【0068】まず、GETI命令を利用して即値データ

10

20

30

40

50

を取出せるようにするには、コンパイルの際に、SKIP 命令により nop 化される命令位置に即値データを格納しておく必要がある。

【0069】図10は、主メモリ10上に、SKIP (2)、定数#1、定数#2、GETI (2, r) の命令列 (定数#1, #2も命令として見なす) が格納されている場合に、SKIP (2) から順に1命令ずつ実行され、GETI (2, r) が実行された際の様子を説明するためのものである。この図10の例では、SKIP (2) でnop化される2つの命令部分に、それぞれ定数#1と定数#2の2つの即値データが予め格納されている。

【0070】図10の例では、SKIP (2)、定数#1、定数#2、GETI (2, r) がSKIP (2) から順に1命令ずつ命令レジスタ2にフェッチされる毎に、書込み先ポインタ42がデクリメントされて、そのデクリメント後の当該ポインタ42の指定する循環バッファ41内エントリに、上記フェッチされた命令とそのアドレスが記録される (ステップS1)。また、SKIP (2) の次の2つの定数#1, #2は、当該SKIP (2) の指定によりnop化される (ステップS2, S3)。したがって、SKIP (2) の次の2命令部分に、即値データである定数#1, #2が格納 (設定) されていても、何ら問題はない。

【0071】さて、GETI (2, r) が命令レジスタ2にフェッチされて、当該GETI (2, r) とそのアドレスが、図10に示すように、書込み先ポインタ42の指す循環バッファ41内エントリに記録され (ステップS1)、更に当該GETI (2, r) が命令デコーダ8でデコードされたものとする (ステップS4)。

【0072】この場合、図6を参照して説明したことからも明かなように、GETI (2, r) のiフィールドの値「2」(i=2)と書込み先ポインタ42の値との (加算器43による) 加算値で指定される循環バッファ41内エントリから読出される命令およびアドレスの対、即ち現在書込み先ポインタ42の指しているエントリ中の命令GETI (2, r) に対して2サイクル前の命令およびアドレスの対のうちの命令 (ここでは、命令に代えて格納されている定数#1) が、図10に示すように、当該GETI (2, r) のrフィールドの指定する、レジスタファイル7内のrレジスタに格納される (ステップS5)。

【0073】このように、SKIP (2) によりnop化される2命令部分に即値データである定数#1, #2を格納しておいた場合、定数#2の次の位置のGETI (2, r) の実行により、定数#1 (即値データ) を (レジスタからのデータロードを指定する) ロード命令を行うことなく、取得できる。なお、GETI (2, r) に代えてGETI (1, r) を用いるならば、定数#2を取得できることは勿論である。

【0074】次に、図1の演算処理装置において、SKIP 命令 (指定命令数nop化命令) でnop化された命令部分 (位置) に引数を格納しておき、当該引数をGETI 命令 (命令履歴取出し命令) を利用してレジスタファイル7内に取出す場合 (固定引数引渡し) につき、図11を参照して説明する。

【0075】まず、GETI 命令を利用して、引数 (固定引数) を引渡せるようにするには、コンパイルの際に、SKIP 命令によりnop化される命令位置に関数 (手続き呼出し) の引数を格納し、更に飛び先にGETI 命令を挿入しておく必要がある。

【0076】図11は、主メモリ10上に、SKIP (2)、引数arg1、引数arg2、手続き呼出し命令であるCALL (FN1) の命令列 (引数arg1, arg2も命令として見なす) が格納され、更にCALL (FN1) の指定する飛び先FN1 (に対応するアドレス) にGETI (2, r) が格納されている場合に、SKIP (2) から順に1命令ずつ実行され、CALL (FN1) の実行の後、その飛び先のGETI (2, r) が実行された際の様子を説明するためのものである。この図11の例では、SKIP (2) でnop化される2つの命令部分に、それぞれ固定の引数arg1と引数arg2が予め格納されている。

【0077】図11の例では、SKIP (2)、引数arg1、引数arg2、CALL (FN1)、GETI (2, r) がSKIP (2) から順に1命令ずつ命令レジスタ2にフェッチされる毎に、書込み先ポインタ42がデクリメントされて、そのデクリメント後の当該ポインタ42の指定する循環バッファ41内エントリに、上記フェッチされた命令とそのアドレスが記録される (ステップS1)。また、SKIP (2) の次の2つの引数arg1, arg2は、当該SKIP (2) の指定によりnop化される (ステップS2, S3)。したがって、SKIP (2) の次の2命令部分に、引数arg1, arg2が格納 (設定) されていても、何ら問題はない。

【0078】さて、GETI (2, r) が命令レジスタ2にフェッチされて、当該GETI (2, r) とそのアドレスが、図11に示すように、書込み先ポインタ42の指す循環バッファ41内エントリに記録され (ステップS1)、更に当該GETI (2, r) が命令デコーダ8でデコードされたものとする (ステップS4)。

【0079】この場合、図6を参照して説明したことからも明かなように、GETI (2, r) のiフィールドの値「2」(i=2)と書込み先ポインタ42の値との (加算器43による) 加算値で指定される循環バッファ41内エントリから読出される命令およびアドレスの対、即ち現在書込み先ポインタ42の指しているエントリ中の命令GETI (2, r) に対して2サイクル前の命令およびアドレスの対のうちの命令 (ここでは、命令

に代えて格納されている引数 $arg\ 2$) が、図11に示すように、当該GETI (2, r) のrフィールドの指定する、レジスタファイル7内のrレジスタに格納される(ステップS5)。

【0080】このように、SKIP (2) によりnop化される2命令部分に関数(手続き呼出し)の引数 $arg\ 1$, $arg\ 2$ を格納し、飛び先にGETI (2, r) を挿入しておいた場合、GETI (2, r) の実行により、引数 $arg\ 2$ を(レジスタからのデータロードを指定する)ロード命令を行うことなく、取得できる。なお、GETI (2, r) に代えてGETI (3, r) を用いるならば、引数 $arg\ 1$ を取得できることは勿論である。

【0081】ところで、スーパーコンピュータなどでは、1つの命令の実行完了を待たずに次の命令を実行可能な構成となっていることが多い。このようなものにおいては、エラートラップが発生しても、そのエラー発生命令のアドレスを取得することは困難である。そこで、図1の演算処理装置において、エラートラップが発生したら、GETA命令(アドレス履歴取出し命令)に飛ぶようにしておき、当該GETA命令を利用してエラー発生命令のアドレスをレジスタファイル7内に取出す場合(エラー発生命令のアドレス取出し)につき、図12を参照して説明する。

【0082】まず、GETA命令を利用して、エラー発生命令のアドレスを取出せるようにするには、コンパイルの際に、エラートラップしたらGETA命令に飛ぶようにしておく必要がある。また、条件処理を必要とする場合には、SKIP命令を利用して分岐なしで条件処理が行えるようにしておく。

【0083】図12は、主メモリ10上に、例えば3サイクルで実行結果が生成されるパイプライン演算命令である浮動小数点加算命令 $fadd$ と、後続する何らかの2命令とが格納され、更に当該2命令のうちの後の命令の実行中に先行する命令 $fadd$ がエラートラップした場合の割込み先にGETA (3, r) が格納されている場合に、命令 $fadd$ から順に1命令ずつ実行が開始され、命令 $fadd$ がエラートラップした結果、GETA (3, r) が実行された際の様子を説明するためのものである。なお、命令 $fadd$ のアドレスは“1000”であるものとする。

【0084】図12の例では、命令 $fadd$ 、後続の2命令が1命令ずつ命令レジスタ2に順次フェッチされる毎に、書込み先ポインタ42がデクリメントされて、そのデクリメント後の当該ポインタ42の指定する循環バッファ41内エンタリに、上記フェッチされた命令とそのアドレスが記録される(ステップS1)。

【0085】ここで、命令 $fadd$ が、実行開始サイクルより2サイクル後にエラートラップしたものとする。この場合、割込み先のGETA (3, r) が命令レジスタ

2にフェッチされると同時に、当該命令 $fadd$ とそのアドレスが、当該命令 $fadd$ の1サイクル前にフェッチされた命令(命令 $fadd$ に後続する2命令のうちの後の命令)の次に位置する循環バッファ41内エンタリに記録される(ステップS1)。そして、命令レジスタ2にフェッチされたGETA (3, r) が命令デコーダ8でデコードされたものとする(ステップS6)。

【0086】この場合、図7を参照して説明したことからも明らかのように、GETA (3, r) のiフィールドの値「3」($i=3$)と書込み先ポインタ42の値との(加算器43による)加算値で指定される循環バッファ41内エンタリから読出される命令およびアドレスの対、即ち現在書込み先ポインタ42の指しているエンタリ中の命令GETA (3, r) に対して3サイクル前のエラー発生命令 $fadd$ およびアドレスの対のうちのアドレス(ここでは“1000”)が、図12に示すように、当該GETI (3, r) のrフィールドの指定する、レジスタファイル7内のrレジスタに取出される(ステップS7)。

【0087】次に、図1の演算処理装置において、CALL命令(手続き呼出し命令)の飛び先にGETA命令を挿入すると共に、手続き終了位置にRET命令(リターン命令)を挿入しておき、GETA命令を利用してCALL命令のアドレスをレジスタファイル7内に取出す場合(手続き呼出し時の戻りアドレス指定)につき、図13を参照して説明する。

【0088】図13は、主メモリ10上に、CALL (FN1)、当該CALL (FN1) による手続き呼出しから戻った際に実行される命令#1を先頭とする命令列が格納され、更にCALL (FN1) の指定する飛び先FN1 (に対応するアドレス) から始まる領域にGETA (1, r) を先頭命令とし、レジスタファイル7内のrレジスタの内容に1を加えたアドレス(戻りアドレス)に実行を移すためのRET (r+1) を最終命令とする命令列が格納されている場合に、CALL (FN1) から順に1命令ずつ実行が開始され、RET (r+1) の実行により、戻り先の命令#1にリターンした際の様子を説明するためのものである。なお、CALL (FN1)、命令#1のアドレスは、それぞれ“1000”、“1001”であるものとする。

【0089】図13の例では、まずCALL (FN1) が、続いてGETA (1, r) が命令レジスタ2にフェッチされる。この命令フェッチの都度、書込み先ポインタ42がデクリメントされて、そのデクリメント後の当該ポインタ42の指定する循環バッファ41内エンタリに、上記フェッチされた命令とそのアドレスが記録される(ステップS1)。

【0090】ここで、命令レジスタ2にフェッチされたGETA (1, r) が、命令デコーダ8でデコードされると(ステップS6)、図7を参照して説明したことか

らも明らかなように、GETA (1, r) の i フィールドの値「1」(i=1)と書込み先ポインタ42の値との(加算器43による)加算値で指定される循環バッファ41内エントリから読出される命令およびアドレスの対、即ち現在書込み先ポインタ42の指しているエントリ中の命令GETA (1, r) に対して1サイクル前の命令およびアドレスの対のうちのアドレス(ここでは、CALL (FN1) のアドレス“1000”)が、図13に示すように、当該GETA (1, r) の r フィールドの指定する、レジスタファイル7内の r レジスタに格納される(ステップS7)。

【0091】さて、GETA (1, r) に後続する命令列が1命令ずつ順次命令レジスタ2にフェッチされ、やがてRET (r+1) が当該命令レジスタ2にフェッチされたものとする。すると、RET (r+1) とそのアドレスが、書込み先ポインタ42の指定する循環バッファ41内エントリに記録される。

【0092】そして、この命令レジスタ2にフェッチされたRET (r+1) が命令デコード8でデコードされて、そのデコード結果に従って実行される(ステップS6, S8)。このRET (r+1) は、レジスタファイル7内の r レジスタの内容に1を加えたアドレスを戻りアドレスとして、当該アドレスに実行を移すリターン命令である。ここで、レジスタファイル7内の r レジスタには、先のGETA (1, r) の実行により、CALL (FN1) のアドレス“1000”が格納されている。

【0093】したがって、RET (r+1) の実行により、アドレス“1001”の命令、即ちCALL (FN1) の次のアドレスの命令#1にリターンすることになる。具体的には、アドレス“1001”がプログラムカウンタ3にセットされた後、そのアドレス“1001”の命令#1が命令キャッシュ1から命令レジスタ2にフェッチされる。

【0094】このように、CALL命令の飛び先にGETA (1, r) を挿入し、当該GETA (1, r) 命令を利用してCALL命令のアドレスをレジスタファイル7内の指定レジスタ(rレジスタ)に取出すようにすると共に、手続き終了位置に、当該レジスタの示すアドレスの次のアドレスにリターンするためのRET命令を挿入することで、戻りアドレスをリンクレジスタに設定する操作を行わなくても、RET命令により手続き呼出しから戻るときに、CALL命令のアドレスの次のアドレス(手続き呼出しの戻りアドレス)に実行を移すことができる。

【0095】なお、前記実施例では、書込み先ポインタ42がデクリメントされるものとして説明したが、インクリメントされるものであっても構わない。但し、この場合には加算器43に代えて減算器を設け、GETI (i, r) またはGETA (i, r) 命令の実行時に、この減算器により書込み先ポインタ42の値からGET

I (i, r) またはGETA (i, r) 命令中の i フィールドの値「i」を減算することで、取出しの対象となる循環バッファ41内エントリを指すエントリアドレスを求める必要がある。

【0096】また、前記実施例では、命令キャッシュ1から命令レジスタ2にフェッチされた命令が次のサイクルでデコードされるものとして説明したが、処理の高速化のために、命令レジスタ2に代えて命令バッファを設けて命令キャッシュ1から命令バッファへの命令の先取りを行うようにしても構わない。但し、この場合には、命令キャッシュ1から命令バッファにフェッチされた命令がデコードされるまで、何サイクルかの遅延が生じるため、この遅延に同期して、書込み先ポインタ42の内容を遅延させて対応する命令がデコードされるまで保持するための(パイプラインレジスタ群等の)機構が必要となる。そして、GETI (i, r) またはGETA (i, r) 命令の実行では、同命令中の i フィールドの値との演算に、書込み先ポインタ42の値ではなくて、当該機構に保持されている現在デコード中の命令が記録されている循環バッファ41内エントリを指すアドレスを用いるようにする。

【0097】また、前記実施例では、循環バッファ41には命令およびアドレスの対を格納するものとして説明したが、GETA命令を利用しない演算処理装置であるならば命令だけを、GETI命令を利用しない演算処理装置であるならばアドレスだけを格納するようにしても構わない。

【0098】

【発明の効果】以上詳述したようにこの発明によれば、次に列挙する効果を得ることができる。

(1) 指定命令数nop化命令(SKIP(N)命令)で指定された飛ばしたい個数だけ後続の命令をnop化することができるため、分岐なしで条件処理を実現することができる。

(2) 履歴を取得したい命令より i サイクル後に実行される位置に命令履歴取出し命令(GETI(i, r)命令)を用意しておくことで、当該取出し命令が実行された場合に、当該取出し命令で指定された i サイクル前の命令を命令履歴としてバッファ手段から簡単に取出すことができる。

(3) 指定命令数nop化命令(SKIP(N)命令)によりnop化される命令部分に、即値データ(即値オペランドデータ)または手続き呼出しの固定引数を設定しておき、更にその後に命令履歴取出し命令(GETI(i, r)命令)を用意して、これらを実行させることにより、当該取出し命令で指定された i サイクル前の命令部分の即値データまたは固定引数をバッファ手段から簡単に得ることができる。このように、即値データまたは引数をレジスタからロードして得る必要がないため、レジスタからのロードによるメモリのオーバーヘッド、お

よびレジスタを使用することによるコストの増加を防止することができる。

(4) 実行開始サイクルよりNサイクル後にエラートラップの発生する可能性のある命令(対象命令)が実際にエラートラップした場合の飛び先に、サイクル数N+1を指定するアドレス履歴取出し命令(GETA命令)を用意しておくことで、上記対象命令がエラートラップして当該取出し命令に実行が移った場合に、当該取出し命令で指定されたN+1サイクル前の命令のアドレス、即ち上記対象命令のアドレスをエラー発生命令のアドレスとしてバッファ手段から簡単に取出すことができる。

(5) 手続き呼出し命令の飛び先に、サイクル数1を指定するアドレス履歴取出し命令(GETA(i, r)命令)を用意しておくことで、当該取出し命令が呼出された場合に、当該取出し命令で指定された1サイクル前の命令のアドレス、即ち上記手続き呼出し命令のアドレスを、戻りアドレスの1つ前のアドレスとしてバッファ手段から簡単に取出すことができる。したがって、従来のように、戻りアドレスをリンクレジスタに設定しておく必要がなくなる。

【図面の簡単な説明】

【図1】この発明の一実施例に係る演算処理装置の主要部の構成を示すブロック図。

【図2】同実施例で適用される新規な命令のフォーマット図。

【図3】同実施例における処理の流れを図2に示した命令がフェッチされた際の処理を中心に説明するためのフローチャート。

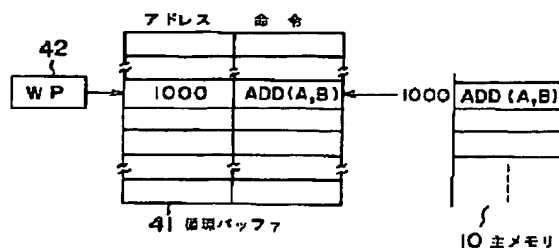
【図4】同実施例におけるバッファ機構4の機能を説明するための図。

【図5】同実施例において指定命令数nop化命令(SKIP命令)を実行することにより実現される指定命令数nop化機能を説明するための図。

【図6】同実施例において命令履歴取出し命令(GETI命令)を実行することにより実現される指定サイクル前の命令の取出し機能を説明するための図。

【図7】同実施例においてアドレス履歴取出し命令(GETA命令)を実行することにより実現される指定サイクル前の命令のアドレスの取出し機能を説明するための図。

【図4】



【図8】同実施例において分岐なしの条件処理を実現するための命令列(モジュール)を、当該条件処理を記述したソースコード、および従来の分岐ありの条件処理となる命令列(モジュール)と対比させて示すもので、図8(a)はソースコード、図8(b)は当該ソースコードのコンパイル結果である従来の命令列、図8(c)当該ソースコードのコンパイル結果である同実施例における命令列。

【図9】同実施例においてGETI命令を利用して、指定サイクル前の命令を取得する命令履歴取出しを説明するための図。

【図10】同実施例においてSKIP命令でnop化された命令部分に即値データを格納しておき、当該即値データをGETI命令を利用してレジスタに取出す即値オペランド指定を説明するための図。

【図11】同実施例においてSKIP命令でnop化された命令部分に引数を格納しておき、当該引数をGETI命令を利用してレジスタに取出す固定引数引渡しを説明するための図。

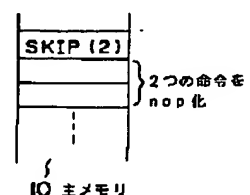
【図12】同実施例においてエラートラップが発生したらGETA命令に飛ぶようにしておき、当該GETA命令を利用してエラー発生命令のアドレスをレジスタに取出すエラー発生命令のアドレス取出しを説明するための図。

【図13】同実施例においてCALL命令(手続き呼出し命令)の飛び先にGETA命令を挿入すると共に、手続き終了位置にRET命令(リターン命令)を挿入しておき、GETA命令を利用してCALL命令のアドレスをレジスタに取出して戻りアドレスの指定に用いるようにした手続き呼出し時の戻りアドレス指定を説明するための図。

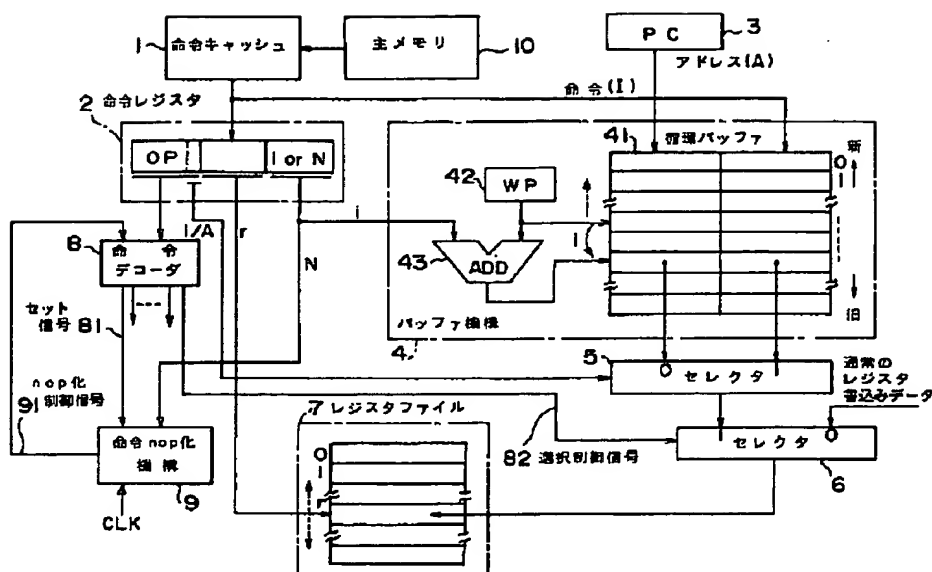
【符号の説明】

1…命令キャッシュ、2…命令レジスタ、3…プログラムカウンタ(PC)、4…バッファ機構、5、6…セクタ、7…レジスタファイル、8…命令デコーダ、9…命令nop化機構、10…主メモリ、41…循環バッファ、42…書き込み先ポインタ(WP)、43…加算器(ADD)、81…セット信号、82…選択制御信号、91…命令nop化制御信号。

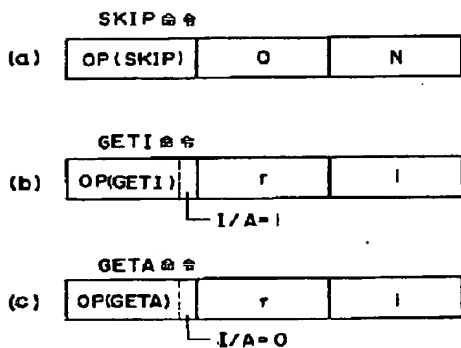
【図5】



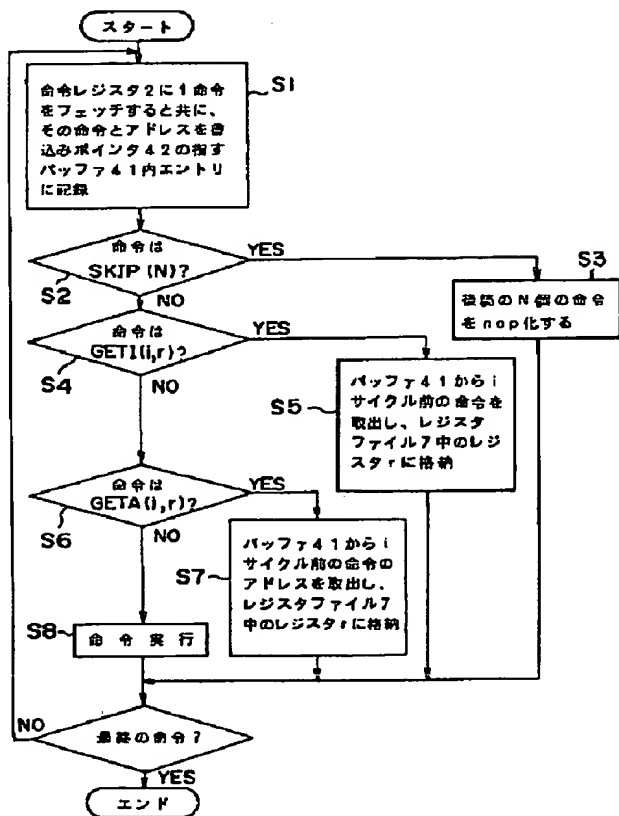
【図1】



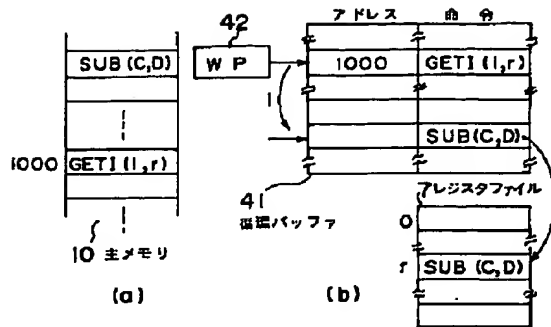
【図2】



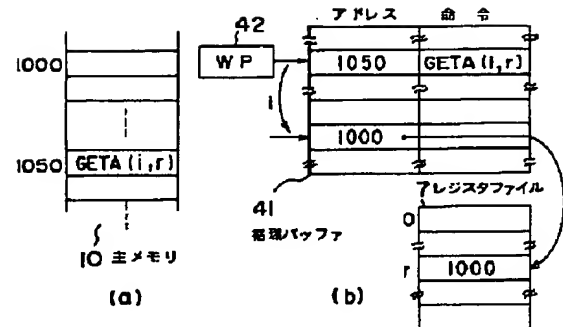
【図3】



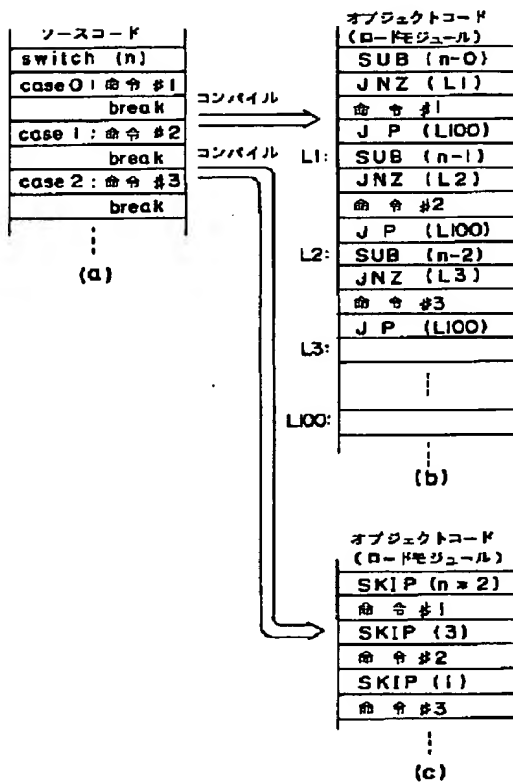
【図6】



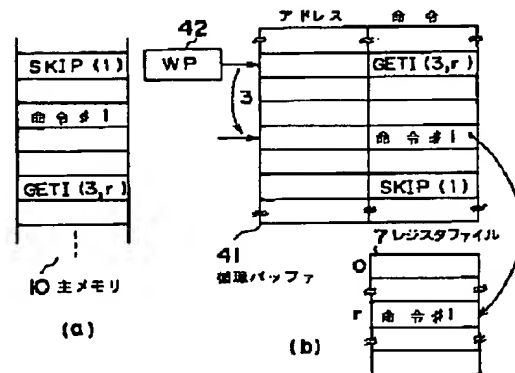
【図7】



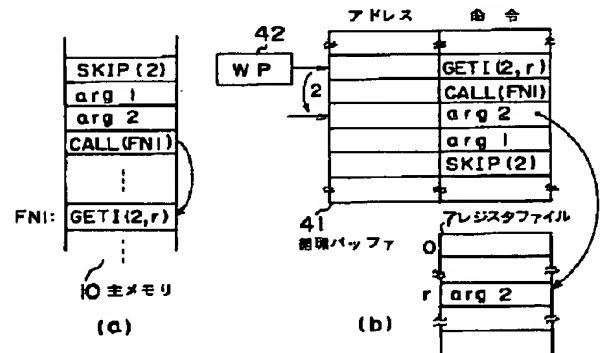
【図8】



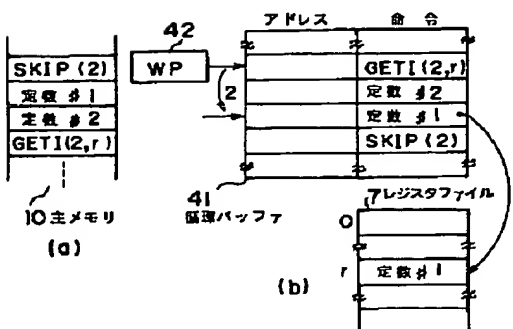
【図9】



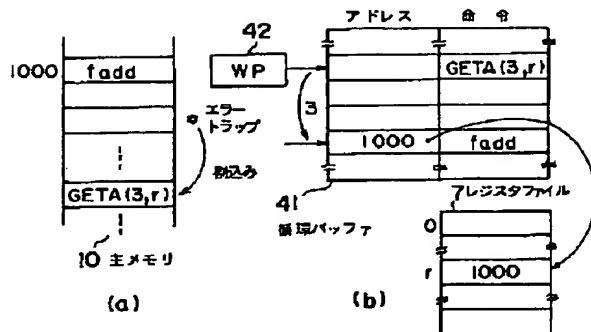
【図11】



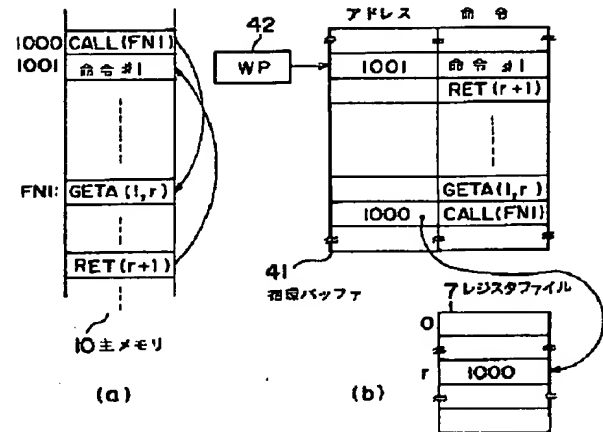
【図10】



【図12】



【図13】



フロントページの続き

(72)発明者 森 良哉
東京都府中市東芝町1番地 株式会社東芝
府中工場内

(72)発明者 武内 和昭
東京都府中市東芝町1番地 株式会社東芝
府中工場内